Lab 2: Comparing Objects, Searching and Sorting

Background

In this assignment, you'll explore ways to sort objects:

- The **Comparable** interface provides a default way to compare two objects of the same type, by implementing the compareTo method.
- The **Comparator** interface allows you to define multiple ways to compare objects, by implementing the compare method.

Note: The **compareTo** method should only be used to sort instances of the same class, while the compare method can be used to compare instances of different classes.

Instructions

1. Working individually, complete the three exercises below using the Lab2-StartingCode provided by your instructor.
2. See the *Marking Criteria* section below for details on how you will be assessed.
3. Submit your completed **zipped** exported Eclipse project to Brightspace by the posted due date.

Exercise 1

1. Implement a class **Student** with the fields **name** and **age**.
2. Implement the Comparable and Comparator interfaces to compare students based on their name and age.
   - The compareTo method, defined by the Comparable interface, compares students based on their name.
   - The compare method, defined by the Comparator interface, compares students based on their age. **If the ages are equal**, it then compares based on their name.
   - **Note:** The Student class implements the Comparable interface, while the Comparator interface will be implemented as an external class to allow for the two different ways to compare students.
3. To test the implementation, use the list of Student objects provided in the exercise1 package and sort it using the Collections.sort method.
4. To ensure the correct functionality, display the list of Student objects with both its name and age before and after it has been sorted.

Exercise 2

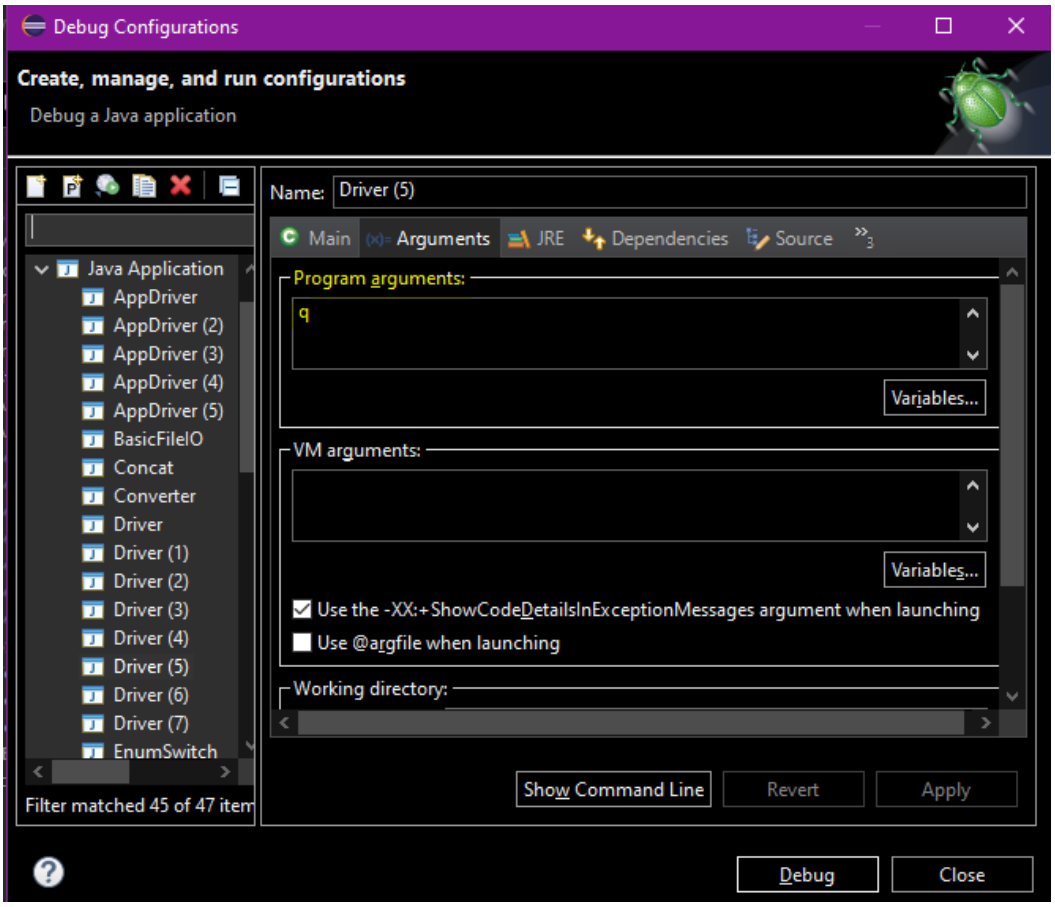
Implement a static method binarySearch that takes the sorted list of integer objects provided in the exercise2 package and an integer target that is prompted from the user when the program first runs, and returns the index of the target in the list if it exists, or -1 if it doesn't. Do NOT use any of the binarySearch() method from the Java library!

Exercise 3

Write a program in Java that sorts an array of integers using **one** of following four different sorting algorithms (Bubble Sort, Insertion Sort, Selection Sort or QuickSort).

- Your program should read the choice of sorting algorithm from the user via command line to choose a sorting algorithm:
    - The valid inputs would be the characters: **b**, **i**, **s** or **q**
    - You can assume that only one single character will be passed into your main method through the command line
        - No error checking is needed for this exercise

You can test this command line in Eclipse using the "Run Configurations" tool under the "Arguments" tab:



- The selected algorithm then sorts the array of integers created in the starting code.
    - Since your program will only have one algorithm implemented, the other 3 will simply do nothing!
- Before and after sorting using any of the sorting algorithms, the program should output the contents of the array in the form of a string.

Marking Criteria

Criteria	**Missing** (0 marks)	**Needs Improvement** (1 mark)	**Good** (2 marks)	**Excellent** (3 marks)	**Marks**
Exercise 1	Not submitted	Significant components are missing	Not all components function as expected	All components are fully functional	**/4**
Exercise 2	Not submitted	Significant components are missing	Not all components function as expected	All components are fully functional	**/3**
Exercise 3	Not submitted	Significant components are missing	Not all components function as expected	All components are fully functional	**/8**
				Total	**/15**